1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	October 1993	Professional Paper

4. TITLE AND SUBTITLE

DETERMINING NEURAL NETWORK HIDDEN LAYER SIZE USING EVOLUTIONARY PROGRAMMING

5. FUNDING NUMBERS

PR: ZW67
PE: 0601152N
WU: DN303002

6. AUTHOR(S)

J. R. McDonnell and D. Waagen

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Naval Command, Control and Ocean Surveillance Center (NCCOSC)
RDT&E Division
San Diego, CA 92152-5001

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Office of Naval Research
800 North Quincy Street
Arlington, VA 22217

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

This work investigates the application of evolutionary programming, a stochastic search technique, for simultaneously determining the weights and the number of hidden units in a fully-connected, multi-layer neural network. The simulated evolution search paradigm provides a means for optimizing both network structure and weight coefficients. Orthogonal learning is implemented by independdently modifying network structure and weight parameters. Different structural level search strategies are investigated by comparing the training processes for the 3-bit parity problem. The results indicate that evolutionary programming provides a robust framework for evolving neural networks.

Published in *Proceedings of the 1993 World Congress on Neural Networks*, Vol. III, pp. 564-567.

14. SUBJECT TERMS

Neural Networks
Evolutionary Programming
Signal Detection

15. NUMBER OF PAGES

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	SAME AS REPORT

21a. NAME OF RESPONSIBLE INDIVIDUAL	21b. TELEPHONE (include Area Code)	21c. OFFICE SYMBOL
J. R. McDonnell	(619) 553-5762	Code 731

Determining Neural Network Hidden Layer Size using Evolutionary Programming

John R. McDonnell & Don Waagen
NCCOSC, RDT&E Div.
San Diego, CA

Abstract

This work investigates the application of evolutionary programming, a stochastic search technique, for simultaneously determining the weights and the number of hidden units in a fully-connected, multi-layer neural network. The simulated evolution search paradigm provides a means for optimizing both network structure and weight coefficients. Orthogonal learning is implemented by independently modifying network structure and weight parameters. Different structural level search strategies are investigated by comparing the training processes for the 3-bit parity problem. The results indicate that evolutionary programming provides a robust framework for evolving neural networks.

Introduction

The traditional neural network design generally consists of a fixed architecture. The static structure determined by the network designer can impose constraints on the network which may not allow for an adequate solution or can result in an unnecessarily complex arrangement with excess parameters. This work investigates the application of evolutionary programming for simultaneously optimizing the network structure as well as the weight coefficients. The network weights are stochastically modified using Gaussian mutations proportional to the network's objective function. Three mutation strategies are investigated in the optimization of the network architecture. The first method incorporates the standard deviation of the hidden unit activation levels over all of the training patterns into the stochastic search process. As a comparison, the second and third methods are purely random approaches.

Evolutionary programming (EP) is a neo-Darwinian, multi-agent, global optimization paradigm that provides a systematic means for employing stochastic search. EP can be applied to the neural network design process to determine both the network structure as well as the network parameters. This investigation focuses on optimizing the number of hidden units in a fully-connected feedforward network. A potential benefit of evolving near minimal size neural networks is that hardware implementations or software emulations may realize a significant increase in throughput for designs which require near real-time capability.

Previous work[1] has utilized EP to reduce the number of connections and, as a result, the number of active hidden units. Others have incorporated variable network structures into the training process. Ash[2] has developed the dynamic node creation algorithm which created new nodes in the hidden layer when the training error rate fell below an arbitrarily chosen critical value. Hirose *et al.*[3] use a similar approach for node creation, but also remove nodes when small error values are attained. Aylward and Anderson[4] postulate a set of rules based on error rate, convergence criterion and distance to the desired error level. If the rules are not continually satisfied, then a new node is added. Fahlman and Lebiere[5] add additional units which are fully connected from previously generated hidden units and the input neurons. Their cascade-correlation architecture selects the best additional hidden unit from a pool of candidate units in minimizing the network error. Lee[6] generates additional neurons based on temporal changes in a neuron's weight space and removes neurons based on activity variance and the distance between similar weight parameter vectors. While not as sophisticated as the techniques used by Lee[6], the approach employed in this work does not require an overt amount of computation for a single network, but instead exploits the stochastic multi-agent search process to evolve a single suitable network from a multitude of candidate networks which constitute a population.

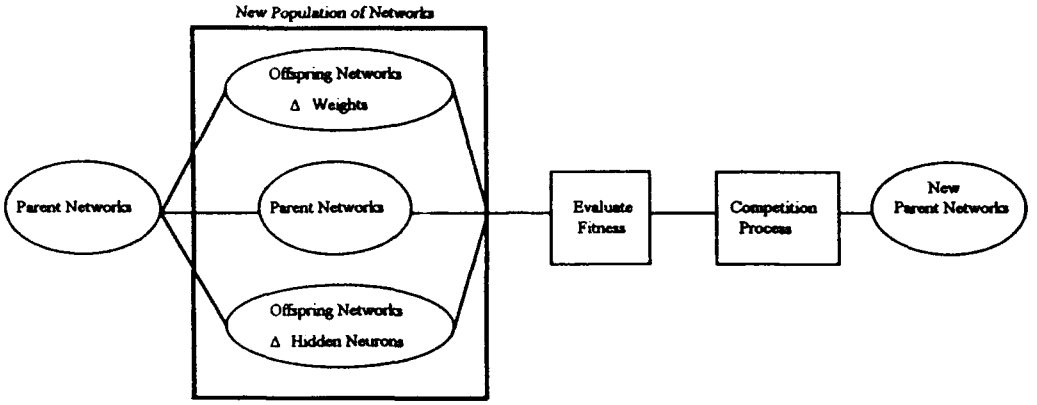

Figure 1. One generation of the evolutionary search process.

Evolving Hidden Layer Size

The EP algorithm utilized in this work is based on the paradigm given by Fogel[7]. A modification was made to account for evolving disparate quantities such as the network weights and the number of hidden units. The following steps describe the EP optimization algorithm employed herein:

1. Form an initial population $P_{3N-1}(w)$ of size 3N. The weight coefficients w associated with each parent element P_i are randomly initialized.
2. Assign a cost J_i to each element $P_i(w)$ in the population based on the objective function.
3. Reorder the population based on the number of wins generated from a stochastic competition process.
4. Generate offspring (P_N .... P_{3N-1}) from the highest ranked N elements (P_0 .... P_{N-1}) in the population. The first set of offspring (P_N .... P_{2N-1}) are generated by perturbing the weights w, and the second set of offspring (P_{2N} .... P_{3N-1}) are generated by the stochastics governing the selection of the hidden nodes (as discussed below).
5. Loop to step 2.

Figure 1 illustrates the evolutionary search process over a single generation. Other techniques (such as backpropagation) could also have been implemented to generate the offspring with a new weight set.

Initially, a population consisting of *3N* feedforward networks is generated. The weights for each network are instantiated from a uniform *U(-0.5, 0.5)* distribution. The maximum number of hidden neurons are selected from a user specified search domain. The objective function *J* is a linearly-weighted combination of the number of hidden nodes N_n and mean sum-squared pattern error *E* and is determined according to $J = \alpha E + \beta N_n$. To emulate the probabilistic nature of survival, a competition process takes place where individual elements compete against randomly chosen members of the population. A network receives a "win" for every comparison with an inferior network (*i.e.*, inferiority equals higher cost). The *N* networks with the most wins are retained and used to generate offspring networks.

The first set of offspring networks are generated simply by perturbing the parent set of weights according to $W_o = W_p + \delta W_p$ where δW_p is a multivariate normally distributed random variable parameterized by $N(0, S_f \cdot J_p)$ with a scaling coefficient S_f and an objective function J_p for each parent network. The scaling factor can be considered the probabilistic analog to the step size used in gradient based methods. A second set of offspring are generated by modifying the number of hidden elements which are active. This is done by activating or deactivating the outputs of randomly selected hidden neurons. Three different strategies to network optimization are investigated in this work. The first method is based on the variance in the activation level of a randomly selected neuron. If the selected hidden node is active, it is deactivated in a probabilistic manner based on the standard deviation σ of the activation level as determined over all input patterns for a given set of weights. The deactivation probability is given by $P_d = 1 - \sigma/\sigma_{max,a}$ where $\sigma_{max,a}$ is the maximum deviation of all the active neurons. If the selected node *i* is inactive, then its probability of being activated is determined by $P_a = \sigma_i / \sigma_{max}$ where σ_{max} =

$\max(\sigma_i, \sigma_{max,a})$ so that $P_a \leq 1$. The second method changes the state of a randomly selected neuron. The third method changes the state of a randomly selected neuron *50%* of the time. Even if a neuron is deactivated, its weights will continue to be updated in the event that the neuron later becomes active. Bias nodes always remain active. After the competition process, the stochastic search procedure is repeated.

In the ensuing investigation, the standard deviation method outlined above is referred to as Strategy 1, the purely random method is referred to as Strategy 2, and the purely random technique with an additional 50% probability of changing a neuron's state of activation is referred to as Strategy 3. Strategy 3 was incorporated to investigate the effects of artificially constraining the search versus the more rigorous search that takes place when a randomly selected neuron's activation state is deterministically changed.

Results

Training trails for the 3-bit parity mapping were conducted for all three strategies. The runs were arbitrarily stopped after *500* generations. Each network was initialized with the maximum *20* hidden nodes fully activated. All of the runs were conducted for *10* parent networks and *20* offspring networks: *10* offspring networks with perturbed weight sets and 10 offspring networks with potential for structural modifications. The objective function is given by $J = E + 0.01*N_n$ and the mutation scaling factor was set as $S_f = 50$. Figure 2 shows the MSE and number of nodes of the network with the best (*i.e.*, lowest cost) J at each generation from a sample training run using Strategy 1. Note that over the first *50* generations the optimization process is primarily reducing the number of hidden nodes. After *90* generations the network has converged to its final form of *3-3-1*.

To evaluate the performance of the three different strategies, comparisons are made for the best network at each generation averaged over ten trail runs. The average MSE of the best network for each strategy is shown in Figure 3. Strategies 1 and 2 converge to roughly the same MSE by 500 generations while Strategy 3, the artificially constrained search, has an average MSE which is 4x greater due to poorer convergence characteristics. The average number of nodes of the best network for each strategy is shown in Figure 4. Both Strategies 1 and 2 converge to approximately the same number of active nodes by 150 generations. The average cost of the best network for each of the strategies is shown in Figure 5. Strategies 1 and 2 are very similar above 150 generations while Strategies 2 and 3 are roughly the same below 100 generations. There appears to be a short transition phase over which the Strategy 2 search reconnects neurons which are significant to the networks performance. The Strategy 3 search is slower to achieve similar results due to the artificial constraints imposed on modifying the network structure. The variance based search does not place an excess emphasis on the number of active neurons and is (on average) primarily disconnecting hidden units.

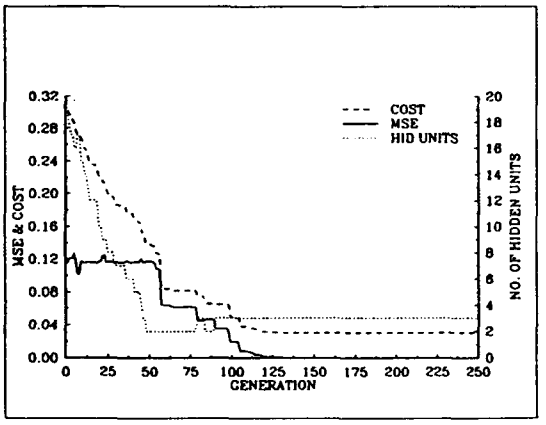

Figure 2. A sample training trial using Strategy 1.

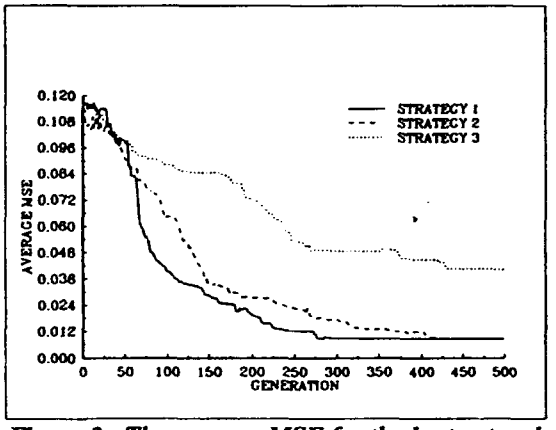

Figure 3. The average MSE for the best network at each generation.

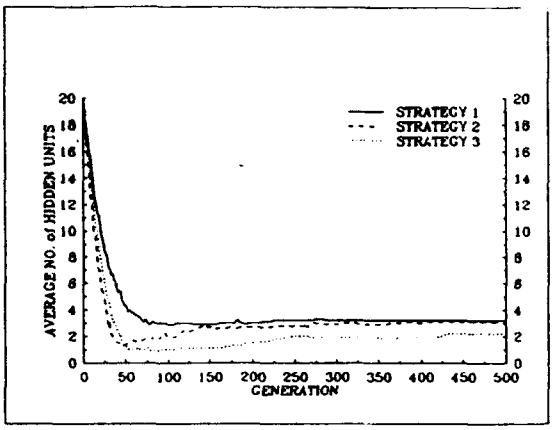

Figure 4. The average number of hidden units for the best network at each generation.

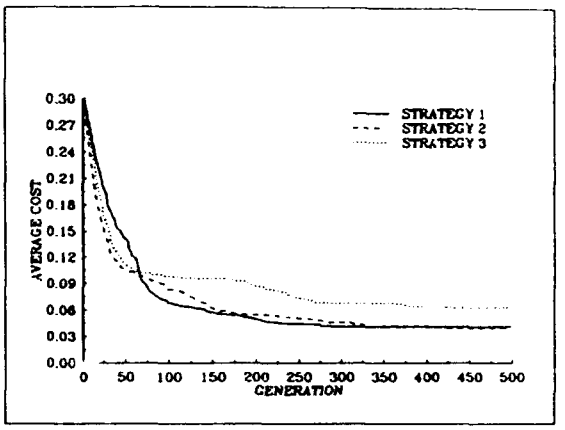

Figure 5. The average cost of the best network at each generation for $J = E + 0.01N_c$.

Conclusion

Although a small sample size was used to determine the performance characteristics using three different network structure level modification strategies, the results appear to indicate that artificially constraining the evolutionary search process impedes the construction of superior networks. Incorporating knowledge into the search process may be advantageous in directing the search, but also comes with added computational cost which was not accounted for in this investigation. In contrasting the purely random approach with the variance-based strategy, there is no significant difference by the final generation. As a result, the simulated evolution paradigm appears to be a robust approach for simultaneously determining the number of hidden nodes and weight coefficients. Further investigations should be undertaken in analyzing the effects of the cost function on stochastic search process.

References

1. J. R. McDonnell and D. Waagen, "Evolving neural network connectivity", *Int. Conf. on Neural Networks*, San Francisco, CA, 1993.

2. T. Ash, "Dynamic node creation in backpropagation networks", *Int. Joint Conf. on Neural Networks*, San Diego, 1989.

3. Y. Hirose, K. Yamashita , and S. Hijiya, "Back-propagation algorithm which varies the number of hidden units", *Neural Networks*, Vol. 4, 1991.

4. S. Aylward and R. Anderson, "An algorithm for neural network architecture generation", *AIAA Computing in Aerospace VIII*, Baltimore, 1991.

5. S. E. Fahlman and C. Lebiere, "The cascade-correlation learning architecture", Technical Report, CMU-CS-90-100, Carnegie Mellon University, Pittsburgh, PA, 1990.

6. T-C. Lee, *Structure Level Adaptation for Artificial Neural Networks*, Kluwer Academic Publishers, 1991.

7. D. B. Fogel, *System Identification through Simulated Evolution: A Machine Learning Approach to Modeling*, Ginn Press, Needham, MA., 1991.